NEOCARE
HEALTH IN A GLANCE

COMPRESSED SENSING

signal processing method that reconstructs a signal from far fewer samples than required by the Nyquist-Shannon sampling theorem

$$Y = \Phi X$$

compressed data

measurement matrix

sparse signal (flattened frame block)

APPROACH

Frame Division –

- Each video frame is split into small square blocks

Sparsification –

- Each block is transformed to a sparse block using Discrete Cosine Transform (DCT).

Compression –

- Random Gaussian measurement matrix (Φ) multiplies flattened block.
- Only a fraction (30%) of the data is kept

WHY SPARSIFY BEFORE CS?

Compressed sensing works best when signal is sparse.

- Natural signals (video frames) are NOT sparse in their raw form: Most pixel values are nonzero — there is no sparsity.
- But many signals become sparse in a transform domain. For videos, DCT is a popular choice because most of the energy is concentrated in just a few DCT coefficients.

RECONSTRUCTION

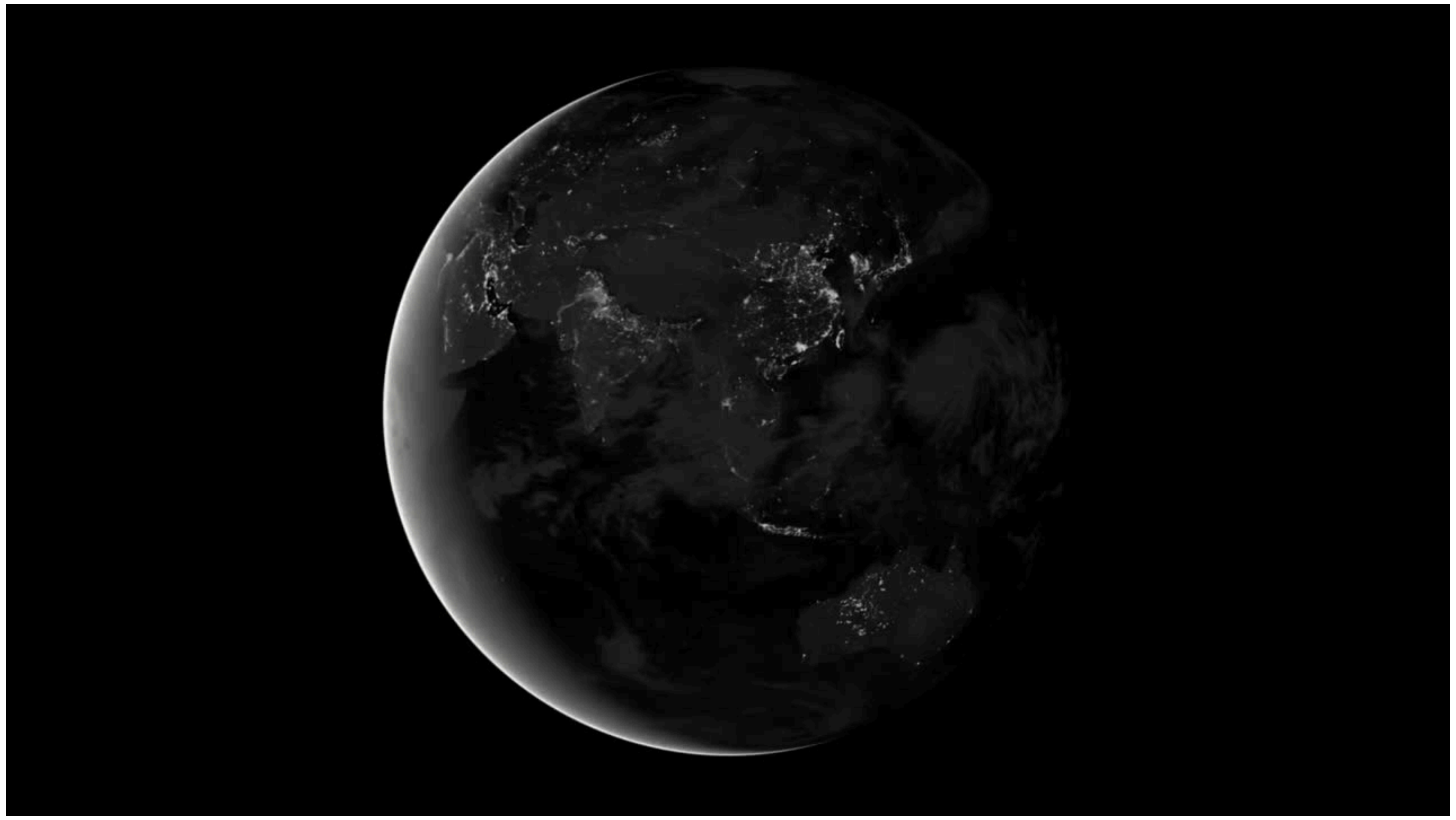
Reconstruction (Sparse Recovery)-

- From the compressed data y, the system recovers x using Orthogonal Matching Pursuit (OMP) - greedy algorithm used for sparse signal recovery. OMP requires the measurement matrix Φ as input.

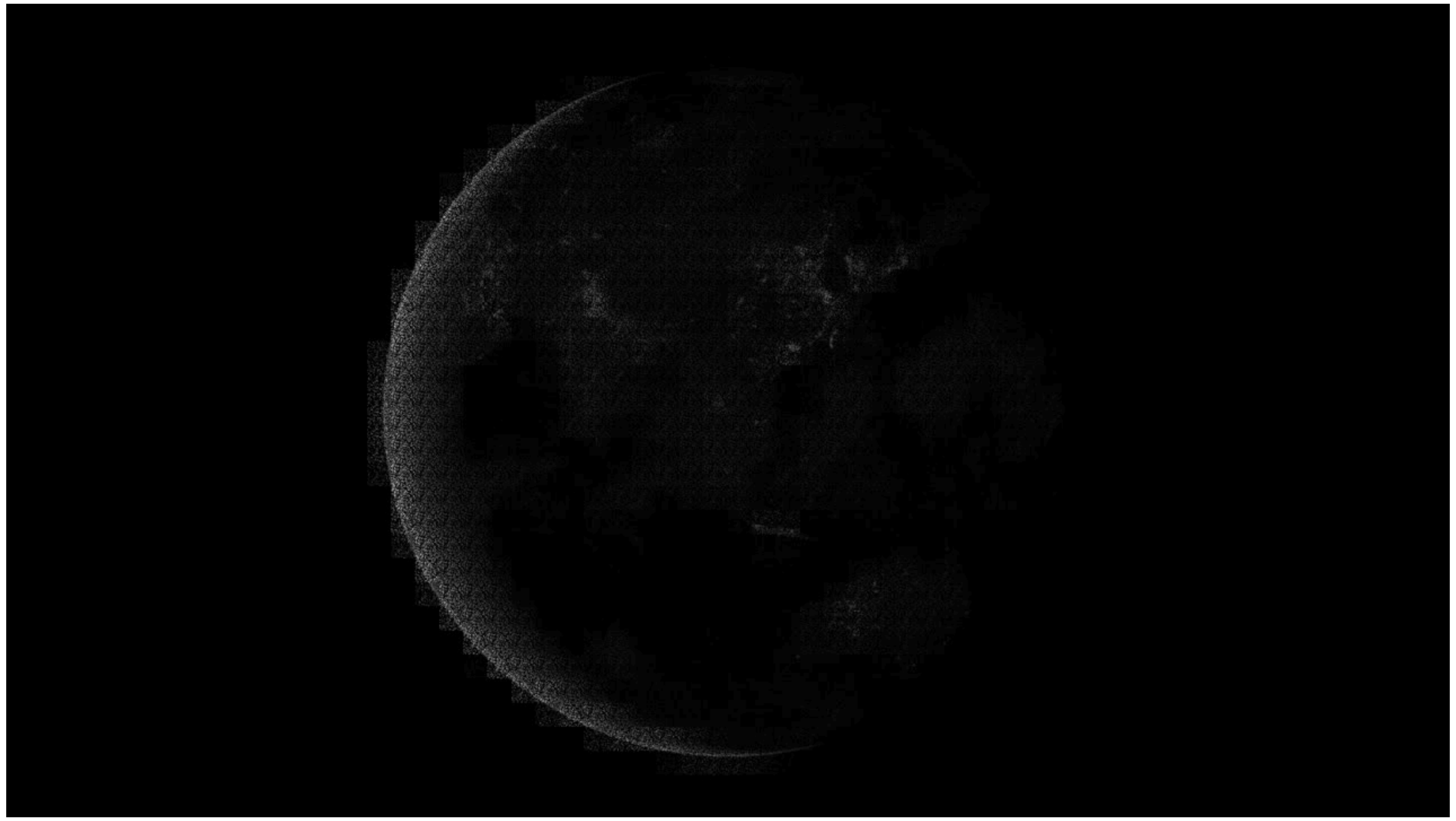
Inverse Transform (IDCT)-

- Recovered sparse coefficients are transformed back using Inverse Discrete Cosine Transform (IDCT) to get the reconstructed spatial domain block

RESULTS



ORIGINAL



RECONSTRUCTED

HOW PRIVACY IS PRESERVED

$$Y=\Phi X$$

- Measurement Matrix as Secret Key:

When we generate Φ randomly and keep it private, only those who have access to Φ can attempt to invert the process and reconstruct x from y.

- Security Through Randomness:

Randomly generated matrices Φ make brute-force or analytic reconstruction infeasible

ISSUES ENCOUNTERED

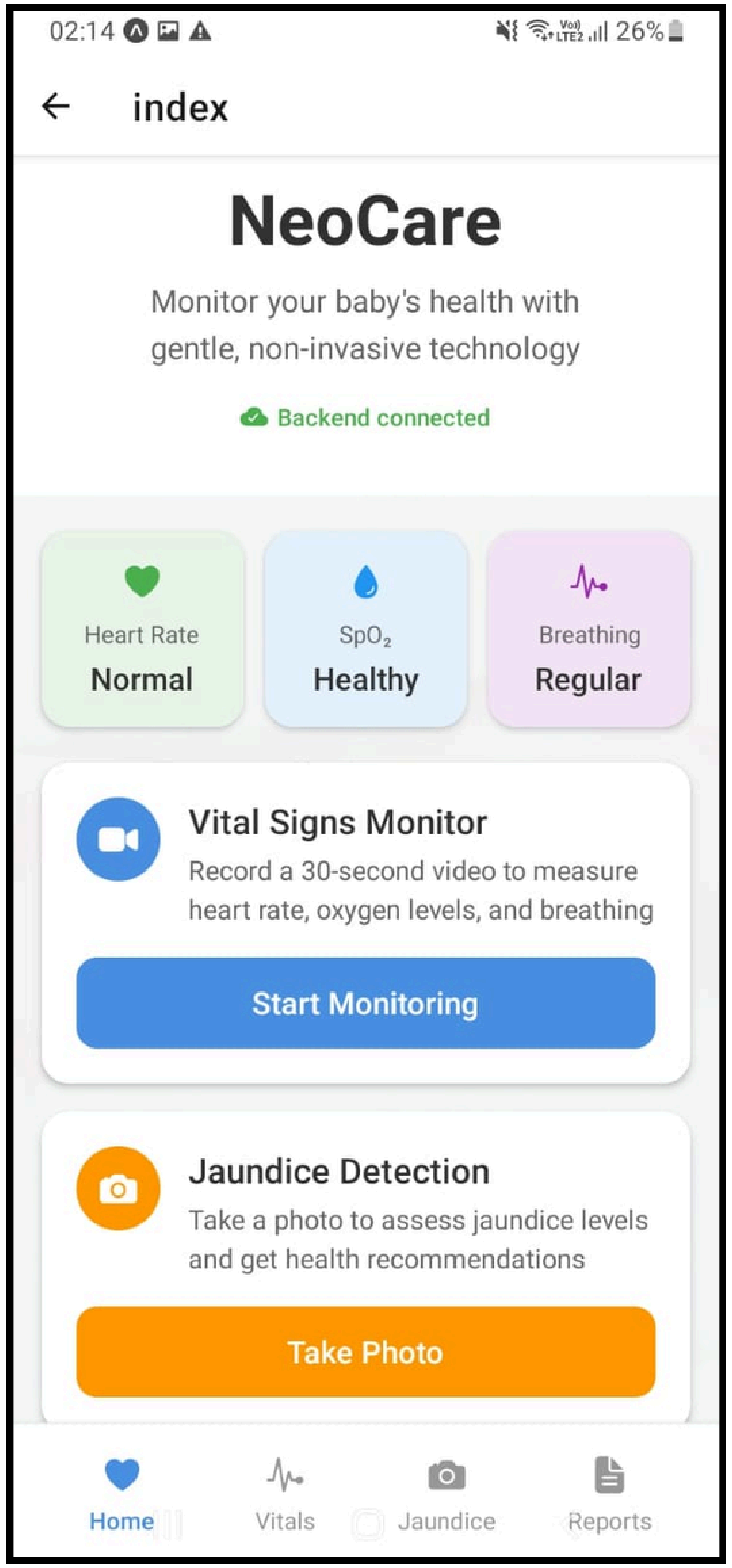
Issue - Measurement matrix is far too large for system memory.

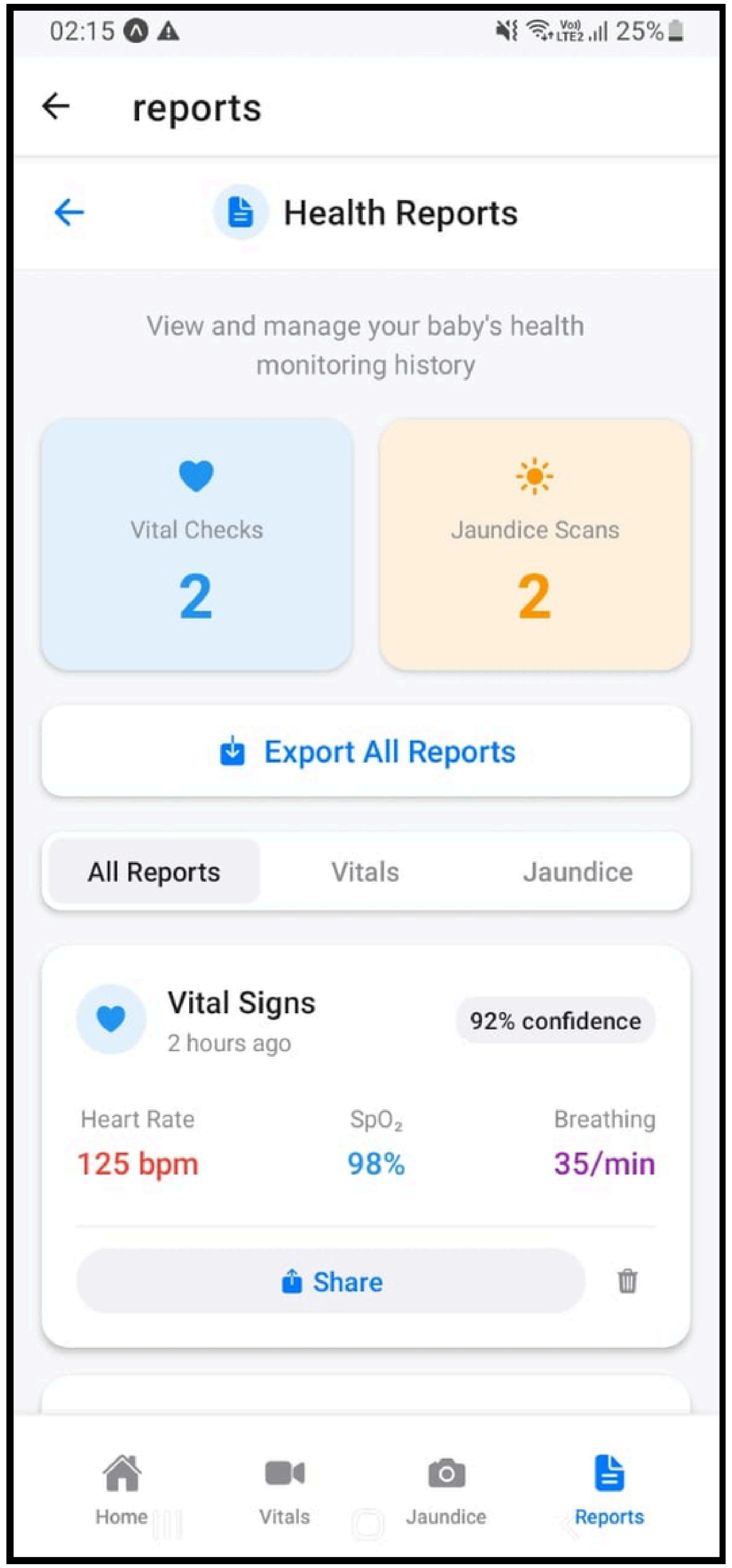
Reason - For a 1920x1080 frame, length of flat frame is 2,073,600. With a compression ratio of 0.3, no of measurements is 622,080. The measurement matrix size is (622,080, 2,073,600), which is huge.

Solution - Using block-based compressed sensing (processing smaller blocks of 32x32 pixels at a time) without creating a measurement matrix for the entire frame.

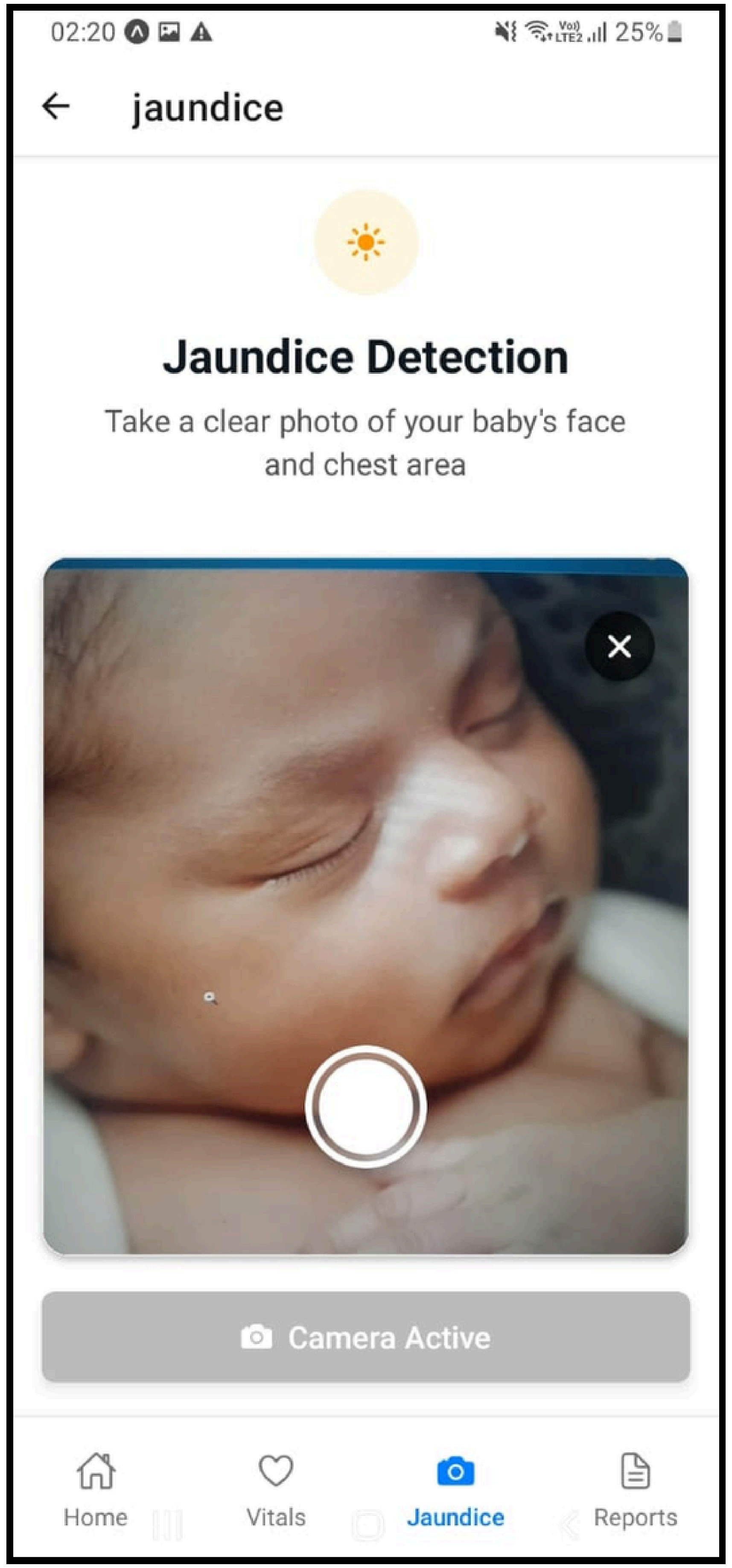
FEATURES IN MOBILE APPLICATION

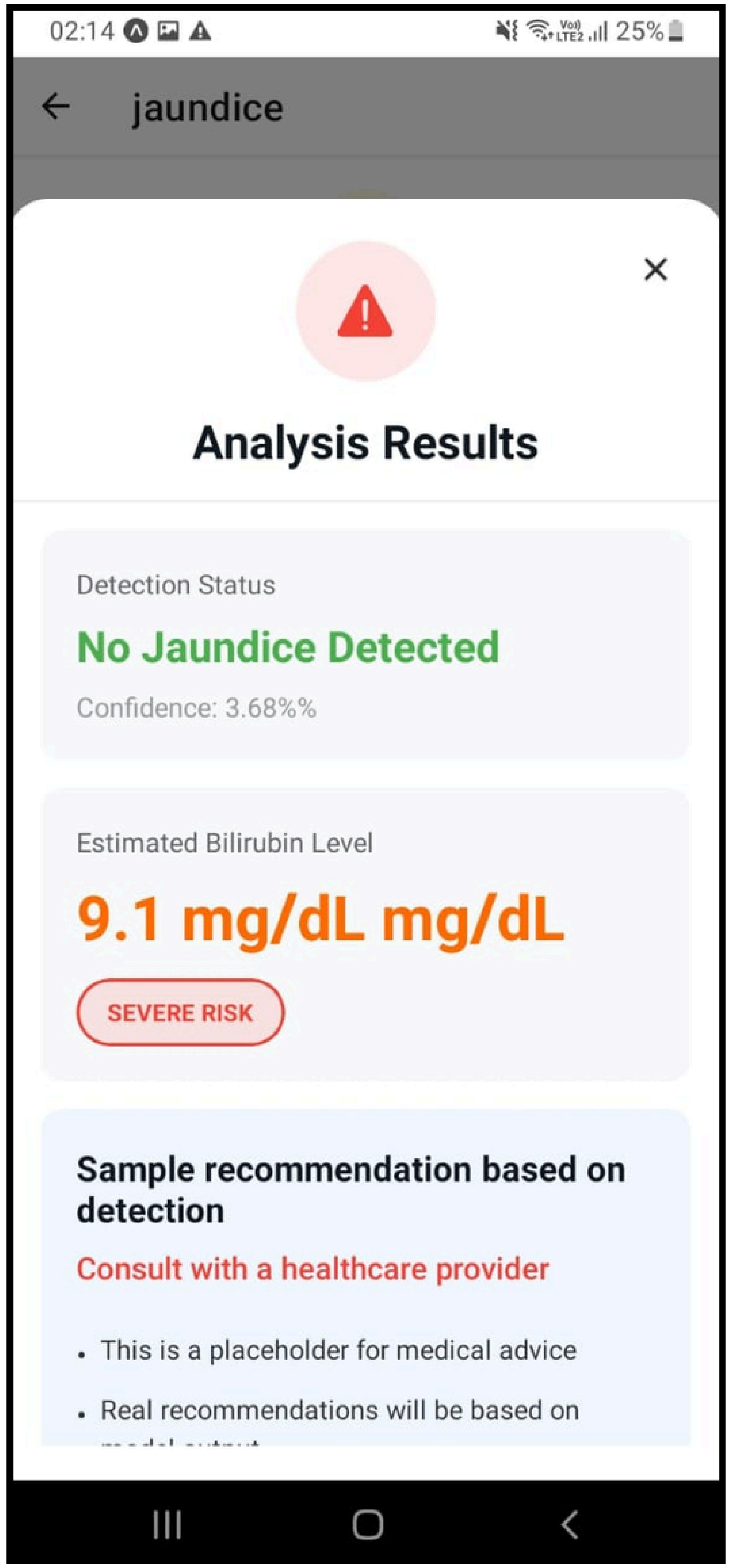
- Created the frontend part using React, using the previously made Figma designs.
- Used Expo - An open-source platform for making universal native apps.
- Test backend using Node.js with Express.js
- It captures video frames using expo-camera





- **Connected the Jaundice detection model with the backend** using RESTful API endpoint that receives images captured by the mobile app, processes them through the pre-trained model and returns jaundice assessment results with confidence scores.

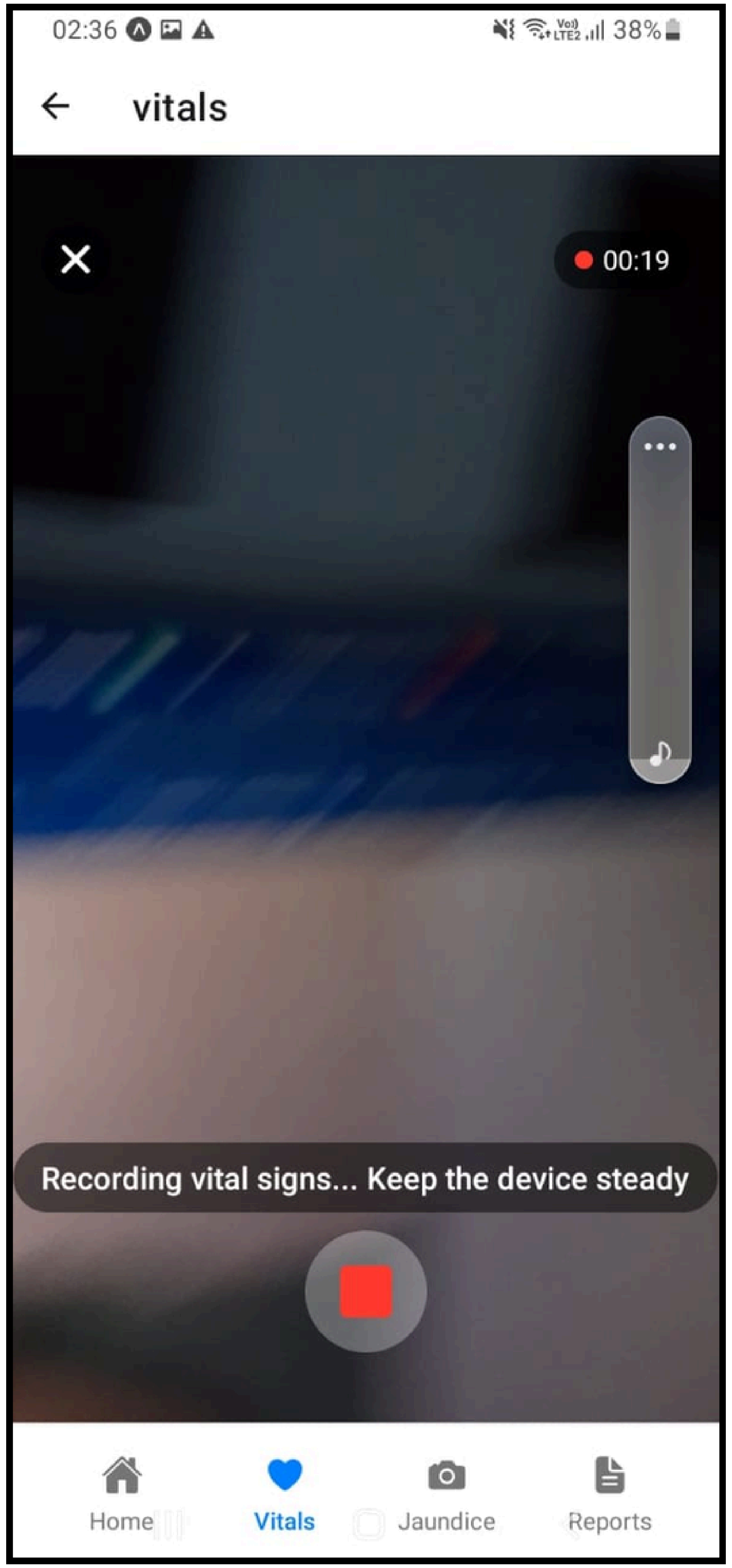


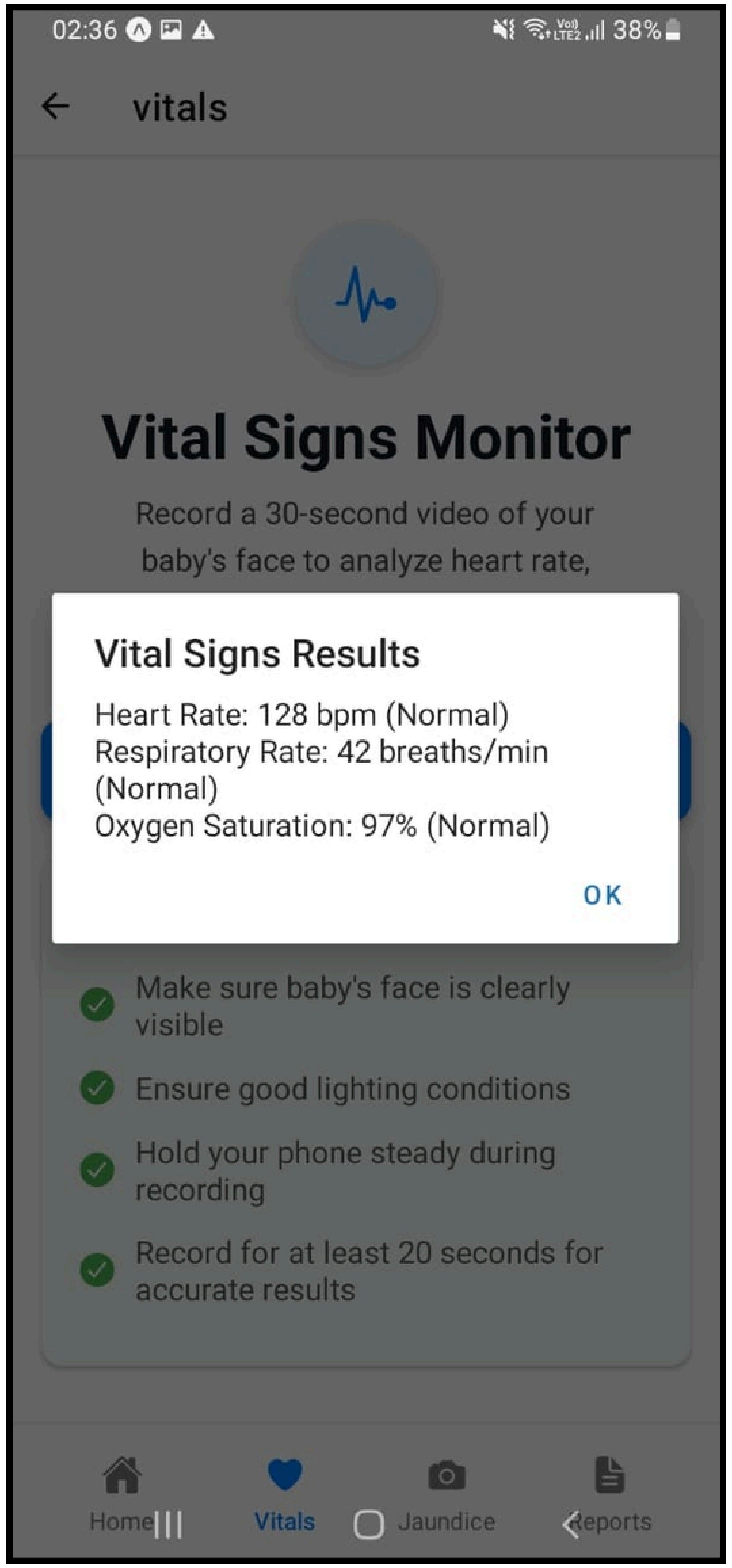


- Connected a test script to detect the other vital signs from the video with a REST API endpoint in local host.

- Implementing multipart video upload with fetch API in React Native, including error and response handling.

- The frontend implementation includes proper video recording cleanup, timer-based recording management.

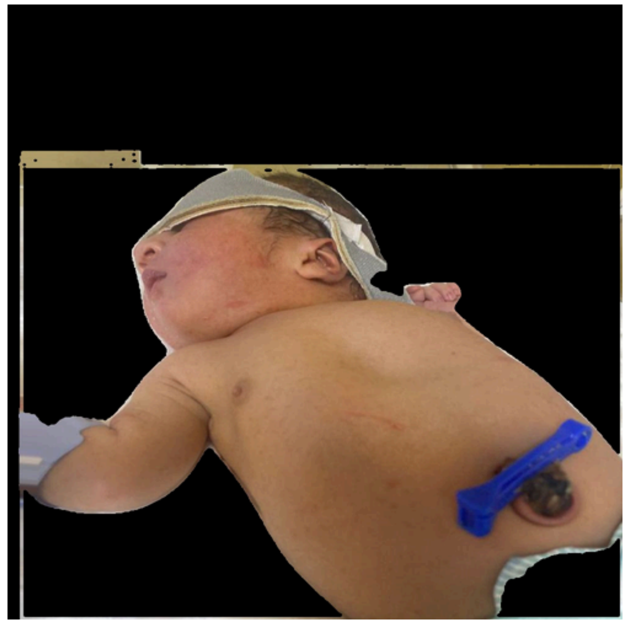




JAUNDICE DETECTION ACCURACY IMPROVEMENT

CREATED SEGMENTED DATASET USING YOLOV8 SEGMENTATION MODEL

- SEGMENTED BABY REGION FROM ORIGINAL IMAGES
- BUILT TRAIN (600 NORMAL + 600 JAUNDICE), VALIDATION (180), AND TEST (180) SETS



DEVELOPED SEPARATE INFERENCE CODE FOR JAUNDICE DETECTION

INPUT IMAGE → **YOLOV8 SEGMENTATION** → **PREPROCESSING (RESIZE + NORMALIZE)** → **TRAINED CNN MODEL (.KERAS)** → **PREDICTION & THRESHOLDING**

USAGE EXAMPLES:

- PYTHON JAUNDICE_INFERENCE.PY /PATH/TO/IMAGE.JPG (***USE LATEST MODEL + 0.5 DEFAULT CLASSIFICATION THERHOLD***)
- PYTHON JAUNDICE_INFERENCE.PY /PATH/TO/IMAGE.JPG --THRESHOLD 0.6 (***USE LATEST MODEL***)
- PYTHON JAUNDICE_INFERENCE.PY /PATH/TO/IMAGE.JPG --MODEL /PATH/TO/MODEL.KERAS

TESTING

CONDUCTED EXPERIMENTS WITH DIFFERENT BATCH SIZES AND LEARNING RATES

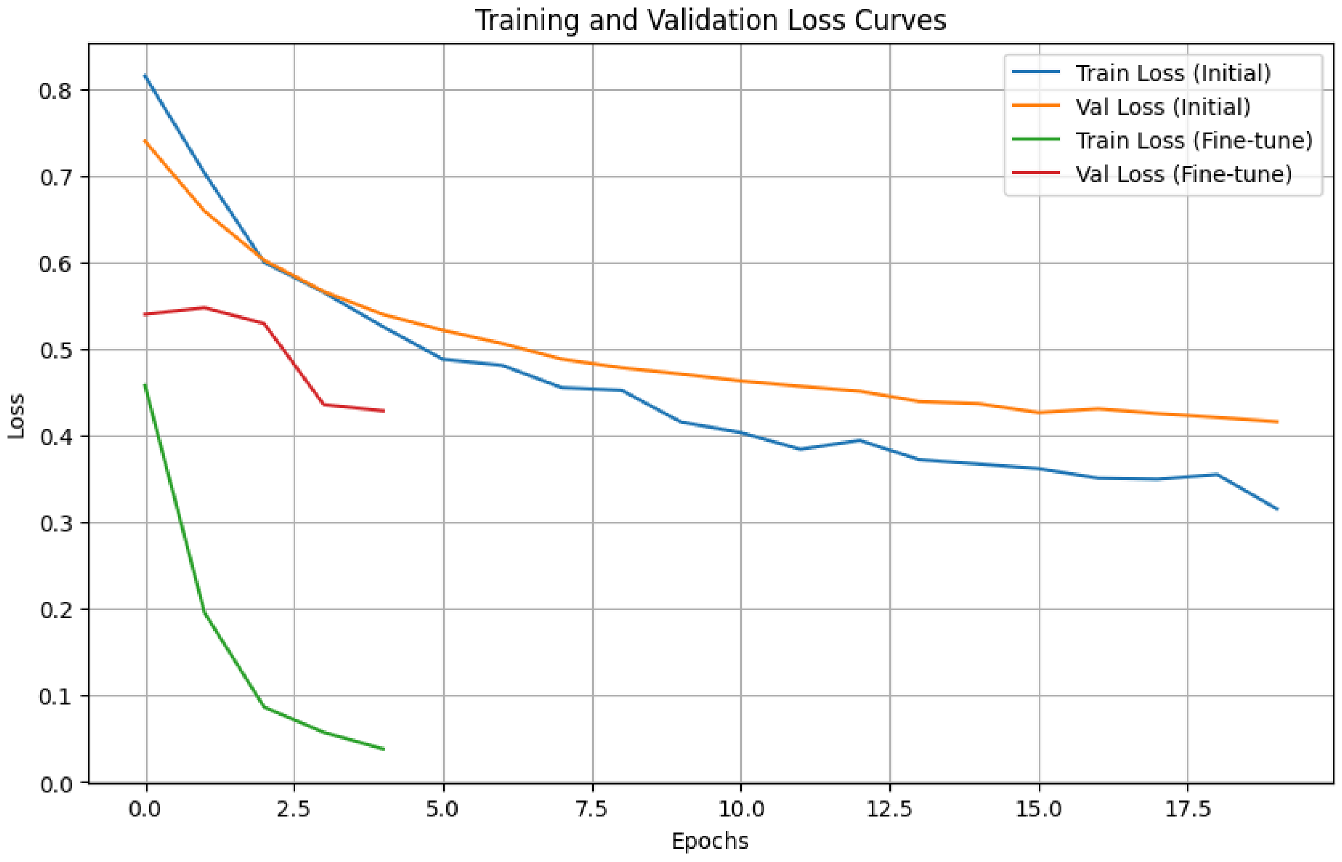
OBSERVED IMPROVEMENT IN ACCURACY AND CONSISTENCY AFTER SEGMENTATION

EXAMPLE:

- BATCH SIZE = 64
- LEARNING RATE = 10E-4

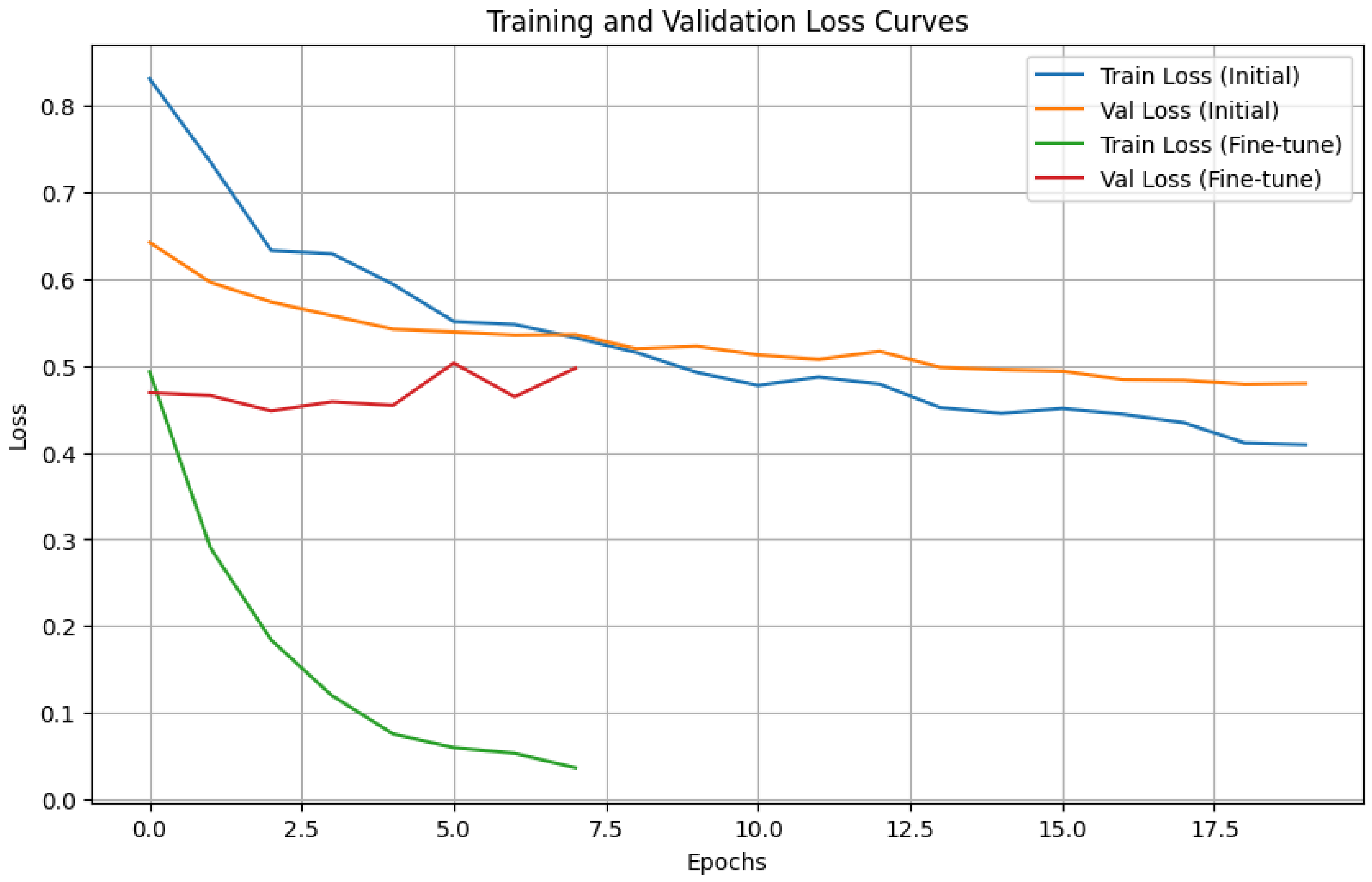
WITHOUT SEGMENTATION

```
Train Accuracy: 82.93%
Validation Accuracy: 78.12%
Test Accuracy: 78.91%
```



USING SEGMENTED DATASET

```
Train Accuracy: 90.50%
Validation Accuracy: 78.12%
Test Accuracy: 77.34%
```



OBSERVATION

MODEL PERFORMS WELL ON TEST DATA FROM THE NJN DATASET, BUT ACCURACY DROPS FOR UNSEEN IMAGES DUE TO:

- BACKGROUND BIAS (NON-BABY REGIONS INFLUENCING PREDICTIONS)
- LIGHTING VARIATIONS AND COLOR TONE DIFFERENCES

UNSEEN TEST IMAGE

TRAINED WITH ROW DATA	TRAINED WITH THE SEGMENTED DATA
=== JAUNDICE DETECTION RESULTS === PREDICTION: NORMAL CONFIDENCE: 90.53% JAUNDICE PROBABILITY: 9.47% NORMAL PROBABILITY: 90.53%	=== JAUNDICE DETECTION RESULTS === PREDICTION: NORMAL CONFIDENCE: 77.66% JAUNDICE PROBABILITY: 22.34% NORMAL PROBABILITY: 77.66%

INFERECNE RESULTS

RESULTS

- SEGMENTED DATASET → NOTICEABLE ACCURACY IMPROVEMENT
- IMPROVED FEATURE FOCUS ON SKIN REGIONS
- REDUCTION OF NOISE FROM BACKGROUND AREAS
- HOWEVER, GENERALIZATION ON UNSEEN DATA STILL LIMITED

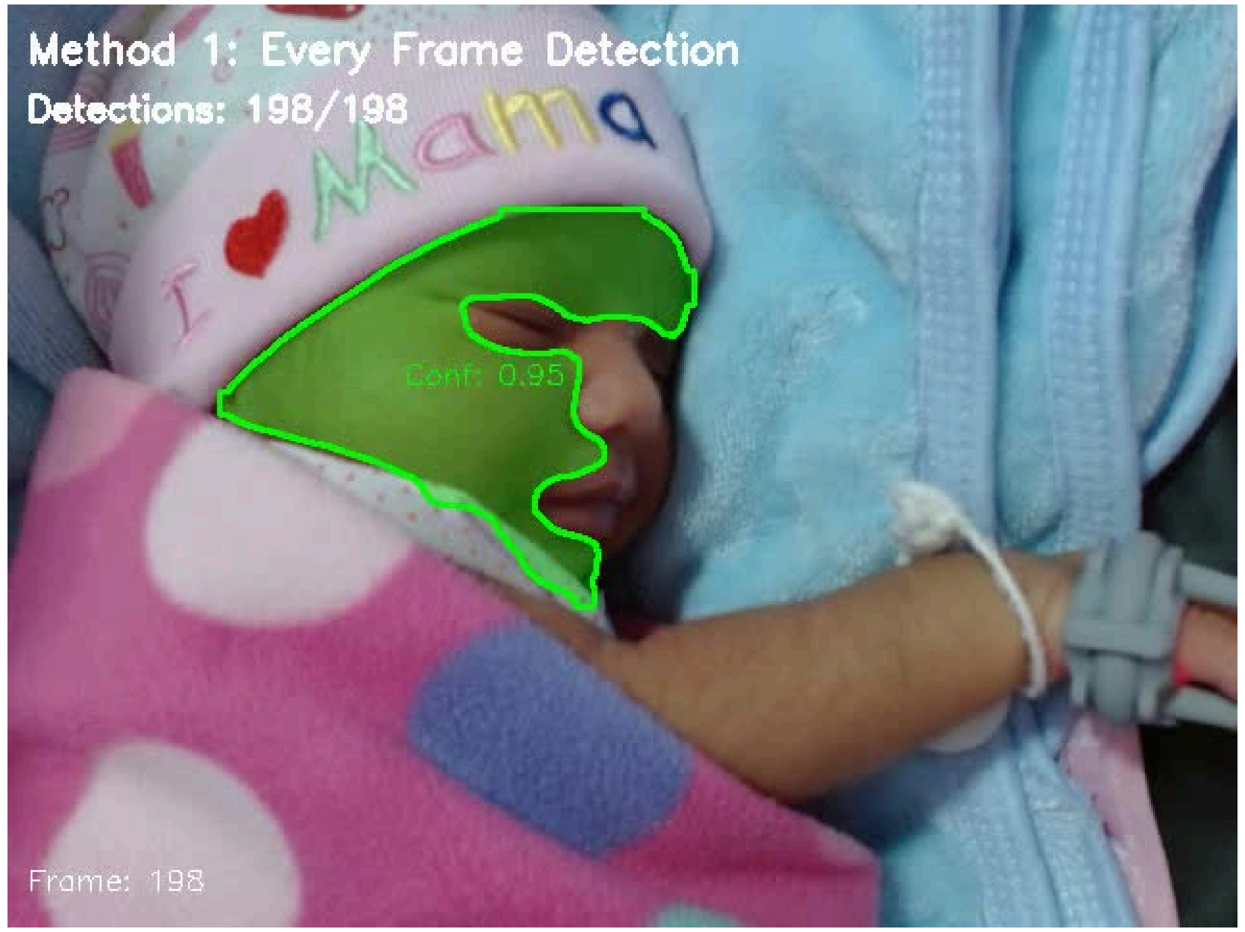
NEXT STEPS

- APPLY COLOR CORRECTION / NORMALIZATION TO HANDLE LIGHTING BIAS
- EXPERIMENT WITH DATA AUGMENTATION FOR UNSEEN SCENARIO ROBUSTNESS
- FINE-TUNE MODEL THRESHOLDS AND EVALUATE ON LARGER DIVERSE DATASETS
- EXPLORE YOLOV9 OR OTHER LIGHTWEIGHT SEGMENTATION MODELS FOR COMPARISON

REGION EXTRACTION IN VIDEOS

METHOD 1 : DETECT EVERY FRAME/ YOLO BASED TRACKING

- Detect region in every frame in the video using the trained model.
- Accuracy is high, but the processing time is large



- Processing speed - 22.9 FPS
- Processing time (for 1 min video) - 89.98 s

- Processing speed - 29.2 FPS
- Speed up - 1.3x

REGION EXTRACTION IN VIDEOS

METHOD 2 : STRICT MOTION FREEZE

- Detect regions in a frame only when motion exceeds a specified threshold, performing detection once and keeping it frozen until new motion occurs with no fallbacks.

Motion threshold	Processing speed	Speed compared to Method 1	Accuracy compared to Method 1 (IOU score)
2%	40.4 FPS	1.7x	0.929
5%	47.8 FPS	2.1x	0.893
10%	57.8 FPS	2.5x	0.904
15%	60.7 FPS	2.6x	0.891

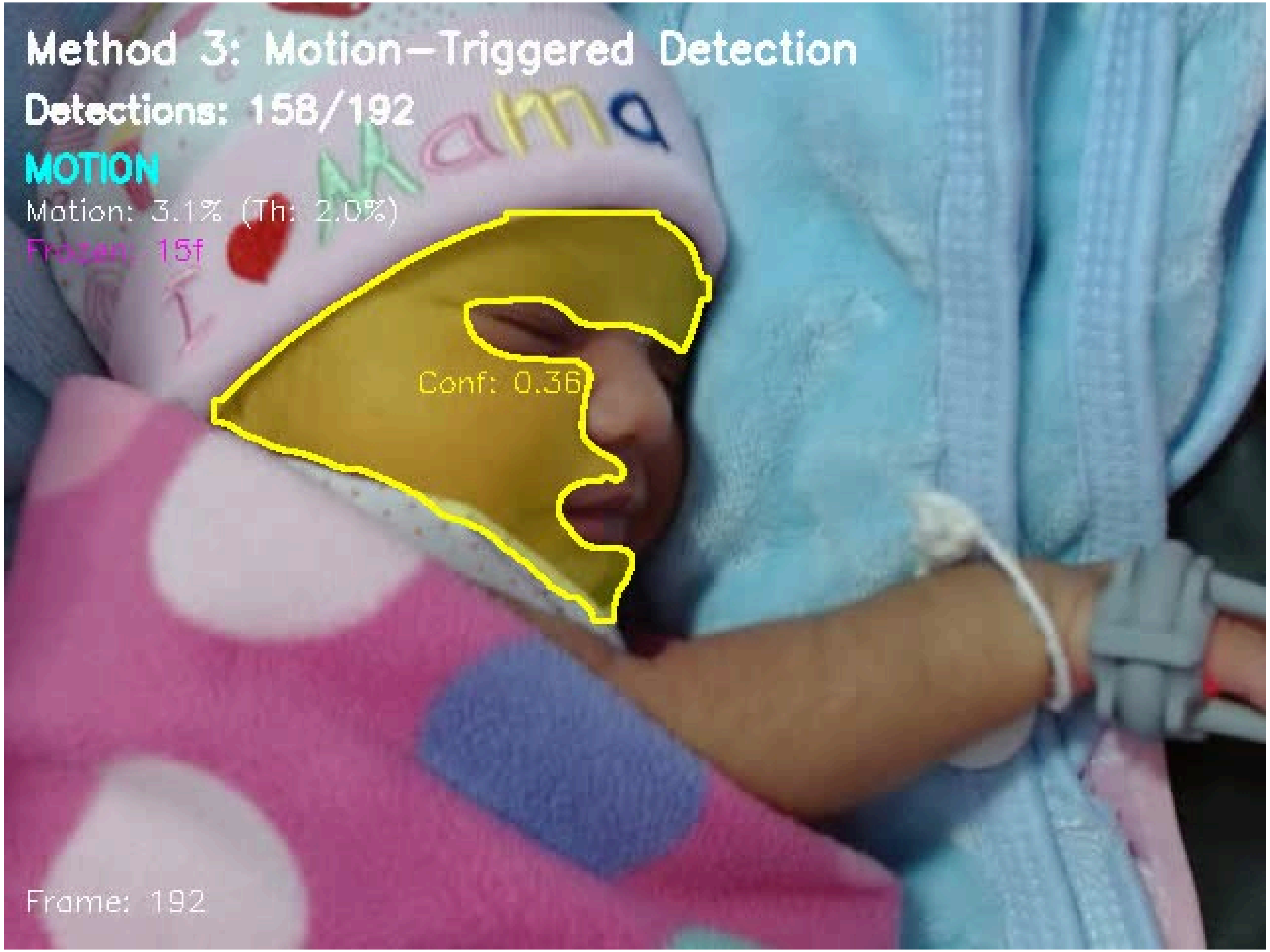

Method 2: Strict Motion Freeze
Detections: 140/192
Status: STRICT FREEZE (29f)
Motion: 0.3% (Th: 2.0%)
Using Frozen Mask
Conf: 0.95
Frame: 192

REGION EXTRACTION IN VIDEOS

METHOD 3 : STRICT MOTION FREEZE WITH FALLBACKS

- Detect regions in a frame only when motion exceeds a specified threshold or when too many frames have been frozen, performing detection once and keeping it frozen until new motion occurs, with fallbacks.

Motion threshold	Processing speed	Speed compared to Method 1	Accuracy compared to Method 1 (IOU score)
2%	39.4 FPS	1.7x	0.976
5%	45.0 FPS	1.9x	0.973
10%	54.7 FPS	2.5x	0.957
15%	59.4 FPS	2.6x	0.953


Method 3: Motion-Triggered Detection
Detections: 158/192
MOTION
Motion: 3.1% (Th: 2.0%)
Frozen: 15f
Conf: 0.36
Frame: 192

REGION EXTRACTION IN VIDEOS

METHOD 4 : INTERVAL BASED DETECTION

- Detect regions in the frames at a given interval length

Interval Length	Processing speed	Speed compared to Method 1	Accuracy compared to Method 1 (IOU score)
10 frames	127.5 FPS	5.5x	0.982
20 frames	169.5 FPS	7.4x	0.974
30 frames	193.1 FPS	8.2x	0.967
60 frames	222.9 FPS	9.6x	0.952


Method 4: Interval 10
Detections: 20/192
Tracking: 172
Conf: 0.95
Frame: 192

PLAN FOR MID EVALUATION

- Train the heart rate and SpO_2 models using the newly segmented data
- Finalize and refine all preprocessing functions.
- Collect a set of the neonatal data from the hospital.